

GLOBAL AI HUB
AYGAZ YAPAY ZEKAYA GİRİŞ PROJE
SUNUMU



Hazırlayan Katılımcının:



Adı: Şevval

Soyadı: ÇETİNKAYA

Mail: sevvall.ctnkaya@gmail.com



Projenin Son Teslim Tarihi: 22.06.2024

Projenin Açıklaması:

Bu proje, Fashion MNIST veri seti üzerinde giysi sınıflandırma problemi için çeşitli makine öğrenimi ve derin öğrenme modellerinin kullanıldığı kapsamlı bir analizdir.

Amaçlar:

- Fashion MNIST veri seti üzerinde bulunan giysi türlerini doğru bir şekilde sınıflandırmak.
- Farklı makine öğrenimi ve derin öğrenme modelleri kullanarak bu sınıflandırma işlemini gerçekleştirmek.
- Modellerin performansını değerlendirmek için çeşitli metrikler kullanmak ve sonuçları görselleştirmek.

Kullanılan Veri Seti

Fashion MNIST, 10 farklı giysi kategorisinden oluşan ve toplamda 70,000 gri tonlamalı 28x28 piksel görüntüden oluşan bir veri setidir. Veri seti, makine öğrenimi ve derin öğrenme algoritmalarının eğitimi ve değerlendirmesi için ideal bir örnek veri setidir.

Kullanılan Teknolojiler ve Yöntemler

- **Python ve Keras:** Model oluşturma, eğitim ve değerlendirme işlemleri için kullanılan programlama dili ve derin öğrenme kütüphanesi.
- **Makine Öğrenimi Modelleri:** K-Nearest Neighbors (KNN), Decision Trees, Random Forest gibi klasik makine öğrenimi algoritmaları.
- **Derin Öğrenme Modeli (Yapay Sinir Ağı):** Sequential model üzerinde katmanlar (Dense, Dropout) kullanılarak yapılandırılan yapay sinir ağı.

Sonuçlar ve Tartışma

- Projede kullanılan makine öğrenimi modelleri (KNN, Decision Trees, Random Forest) ve yapay sinir ağı modeli, Fashion MNIST veri setinde iyi performans sergilemiştir.
- Yapay sinir ağı modeli, diğer modellere göre daha karmaşık yapıları modellemek ve daha yüksek performans elde etmek için tasarlanmıştır.
- Projede elde edilen sonuçlar, giysi sınıflandırma probleminde derin öğrenme modellerinin (YSA) etkinliğini ve önemini göstermektedir.

Bu proje, Fashion MNIST gibi standart bir veri seti üzerinde derinlemesine bir makine öğrenimi ve derin öğrenme uygulaması sunarak, model seçimi, eğitimi, değerlendirmesi ve sonuçlarının yorumlanması konularında geniş bir bakış açısı sağlamaktadır.

Proje Adımları

1. **Kütüphane Yükleme ,Veri Yükleme ve Ön İşleme:** Fashion MNIST veri seti yüklenmiş, görüntüler normalize edilmiş, yeniden şekillendirilmiş ve etiketler kategorik hale getirilmiştir.

```
Aygaz-AI-Project.ipynb ☆
Dosya Düzenle Göster Ekle Çalışma zamanı Araçlar Yardım
Kod + Metin
RAM
Disk
+ Gemini

Kütüphaneleri Yükleme ve Veriyi Hazırlama

[1] import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from keras.datasets import fashion_mnist
from keras.models import Sequential
from keras.layers import Dense, Conv2D, MaxPooling2D, Flatten, Dropout
from keras.utils import to_categorical
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score, confusion_matrix
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from xgboost import XGBClassifier

# Veri setini yükleme
(X_train, y_train), (X_test, y_test) = fashion_mnist.load_data()

# Veri setinin boyutlarını yazdırma
print(f"X_train boyutu: {X_train.shape}")
print(f"y_train boyutu: {y_train.shape}")
print(f"X_test boyutu: {X_test.shape}")
print(f"y_test boyutu: {y_test.shape}")

# Görüntülerin boyutlarını yazdırma
print(f"Bir görüntünün boyutu: {X_train[0].shape}")
```

```
Aygaz-AI-Project.ipynb ☆
Dosya Düzenle Göster Ekle Çalışma zamanı Araçlar Yardım Tüm değişiklikler kaydedildi
Kod + Metin
RAM
Disk
+ Gemini

# Görüntüleri görselleştirme
fig, axes = plt.subplots(1, 10, figsize=(10, 1))
for i in range(10):
    axes[i].imshow(X_train[i], cmap='gray')
    axes[i].axis('off')
plt.show()


# Verileri normalize etme
X_train = X_train / 255.0
X_test = X_test / 255.0

# Veri setini yeniden şekillendirme (Flatten) - Makin öğrenmesi algoritmaları için
X_train_flat = X_train.reshape(-1, 28*28)
X_test_flat = X_test.reshape(-1, 28*28)

# Veri setini yeniden şekillendirme (VSA ve ESA için)
X_train_resaped = X_train.reshape(-1, 28, 28, 1)
X_test_resaped = X_test.reshape(-1, 28, 28, 1)

# Etiketleri kategorik hale getirme
y_train_cat = to_categorical(y_train, 10)
y_test_cat = to_categorical(y_test, 10)

X_train boyutu: (60000, 28, 28)
y_train boyutu: (60000,)
X_test boyutu: (10000, 28, 28)
y_test boyutu: (10000,)
Bir görüntünün boyutu: (28, 28)


```

Proje Adımları

2. ****Makine Öğrenimi Modellerinin Eğitimi ve Değerlendirilmesi:**** KNN, Decision Trees, Random Forest gibi makine öğrenimi modelleri eğitilmiş ve performansları accuracy, precision, recall, F1-score, ROC AUC gibi metrikler kullanılarak değerlendirilmiştir. Confusion matrixler kullanılarak sonuçlar görselleştirilmiştir.

```
Model Eğitim ve Değerlendirme Fonksiyonu

[2] def train_and_evaluate(model, X_train, y_train, X_test, y_test, model_name):
    try:
        model.fit(X_train, y_train)
        y_pred = model.predict(X_test)
        accuracy = accuracy_score(y_test, y_pred)
        precision = precision_score(y_test, y_pred, average='weighted')
        recall = recall_score(y_test, y_pred, average='weighted')
        f1 = f1_score(y_test, y_pred, average='weighted')

        if hasattr(model, "predict_proba"):
            roc_auc = roc_auc_score(y_test_cat, model.predict_proba(X_test), multi_class='ovr')
        else:
            roc_auc = roc_auc_score(y_test_cat, model.predict(X_test), multi_class='ovr')

        cm = confusion_matrix(y_test, y_pred)

        print(f"{model_name} | Accuracy: {accuracy:.4f}, Precision: {precision:.4f}, Recall: {recall:.4f}, F1 Score: {f1:.4f}, ROC AUC: {roc_auc:.4f}")
        return accuracy, precision, recall, f1, roc_auc, cm
    except Exception as e:
        print(f"Model: {model_name} encountered an error: {e}")
        return None, None, None, None, None, None
```

Birden fazla modeli eğitmek ve değerlendirmek çeşitli avantajlar sağlar:

1. Model Performansını Karşılaştırma

Farklı modellerin eğitim ve test setlerindeki performansını karşılaştırarak, hangisinin belirli bir veri seti ve problem için en iyi sonuçları verdiğini belirleyebiliriz. Bu, model seçimi sürecinde kritik bir adımdır.

2. Model Çeşitliliği

Her model farklı varsayımlar ve öğrenme yöntemleri kullanır. Örneğin, KNN komşuluk bilgisine dayanırken Yapay Sinir Ağları (YSA) çok katmanlı öğrenme kullanır. Farklı modelleri denemek, problem için en uygun modelin hangisi olduğunu anlamaya yardımcı olur.

3. Overfitting ve Underfitting

Farklı modellerin overfitting ve underfitting eğilimlerini değerlendirebiliriz. Bazı modeller, eğitim verisine çok fazla uyarak test setinde düşük performans gösterebilir (overfitting), bazıları ise eğitim verisini yeterince öğrenemeyebilir (underfitting). Farklı modelleri denemek, bu dengeyi bulmaya yardımcı olabilir.

4. Model Özellikleri ve Hiperparametreler

Her modelin farklı hiperparametreleri ve özellikleri vardır. Farklı modelleri denemek, hiperparametre ayarlarının ve model özelliklerinin performansı nasıl etkilediğini anlamamızı sağlar.

5. Ensembling ve Model Birleştirme

Farklı modellerin kombinasyonlarını kullanarak ensembling yöntemlerini uygulayabiliriz. Bu, genellikle tek bir modelin performansından daha iyi sonuçlar verir.

6. Güvenilirlik ve Robustluk

Farklı modellerin sonuçlarını karşılaştırmak, model sonuçlarının ne kadar güvenilir olduğunu değerlendirmeye yardımcı olur. Birden fazla model benzer sonuçlar veriyorsa, sonuçlar daha güvenilirdir.

7. Özellik Seçimi ve Mühendisliği

Farklı modeller, farklı özelliklerin önemini belirlememize yardımcı olabilir.

8. Veri Setinin Anlaşılması

Farklı modellerin performansını analiz etmek, veri setinin yapısını ve özelliklerini daha iyi anlamamıza yardımcı olur. Hangi modellerin hangi durumlarda daha iyi performans gösterdiğini görmek, veri setinin zorluklarını ve avantajlarını anlamaya yardımcı olur.

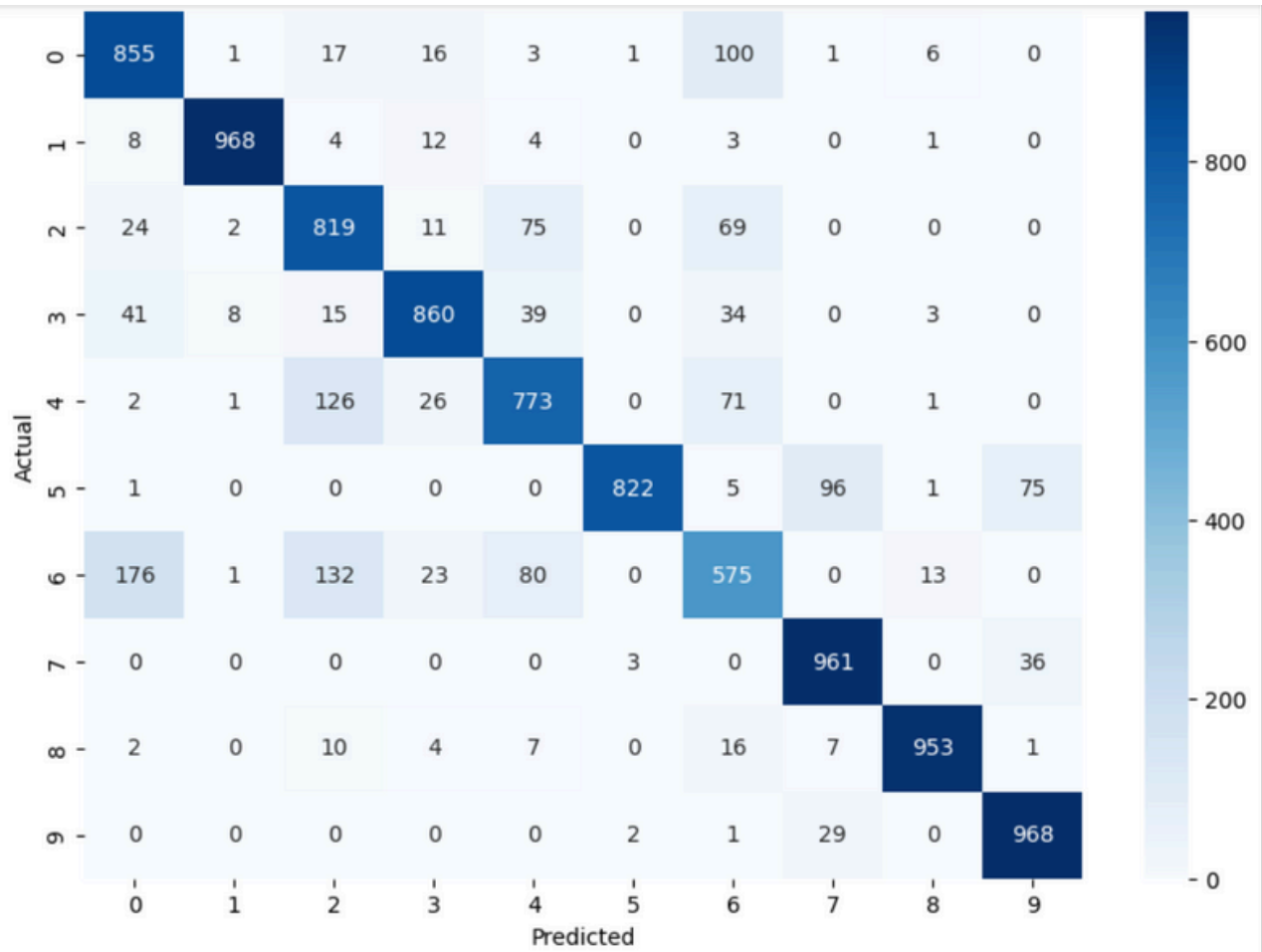
KNN Modeli Eğitimi ve Değerlendirmesi

KNN Modeli

```
knn = KNeighborsClassifier()
accuracy, precision, recall, f1, roc_auc, cm = train_and_evaluate(knn, X_train_flat, y_train, X_test_flat, y_test, 'KNN')

if accuracy is not None:
    plt.figure(figsize=(10, 7))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
    plt.title('Confusion Matrix for KNN')
    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.show()
```

KNN | Accuracy: 0.8554, Precision: 0.8578, Recall: 0.8554, F1 Score: 0.8546, ROC AUC: 0.9685

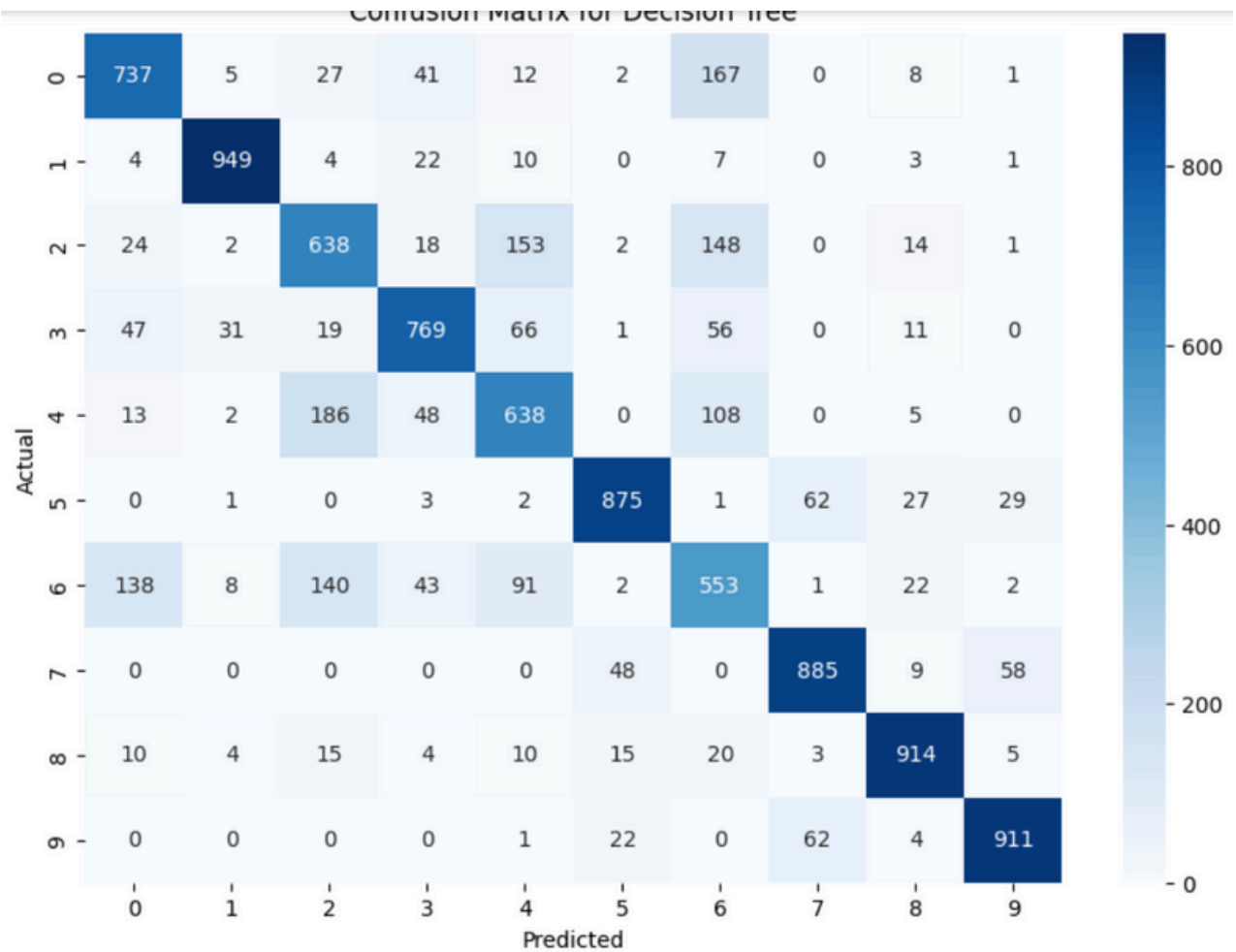


Decision Tree Modeli Eğitimi ve Değerlendirmesi

Decision Tree Modeli

```
decision_tree = DecisionTreeClassifier()  
accuracy, precision, recall, f1, roc_auc, cm = train_and_evaluate(decision_tree, X_train_flat, y_train, X_test_flat, y_test, 'Decision Tree')  
  
if accuracy is not None:  
    plt.figure(figsize=(10, 7))  
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')  
    plt.title('Confusion Matrix for Decision Tree')  
    plt.xlabel('Predicted')  
    plt.ylabel('Actual')  
    plt.show()
```

Decision Tree | Accuracy: 0.7869, Precision: 0.7887, Recall: 0.7869, F1 Score: 0.7877, ROC AUC: 0.8816



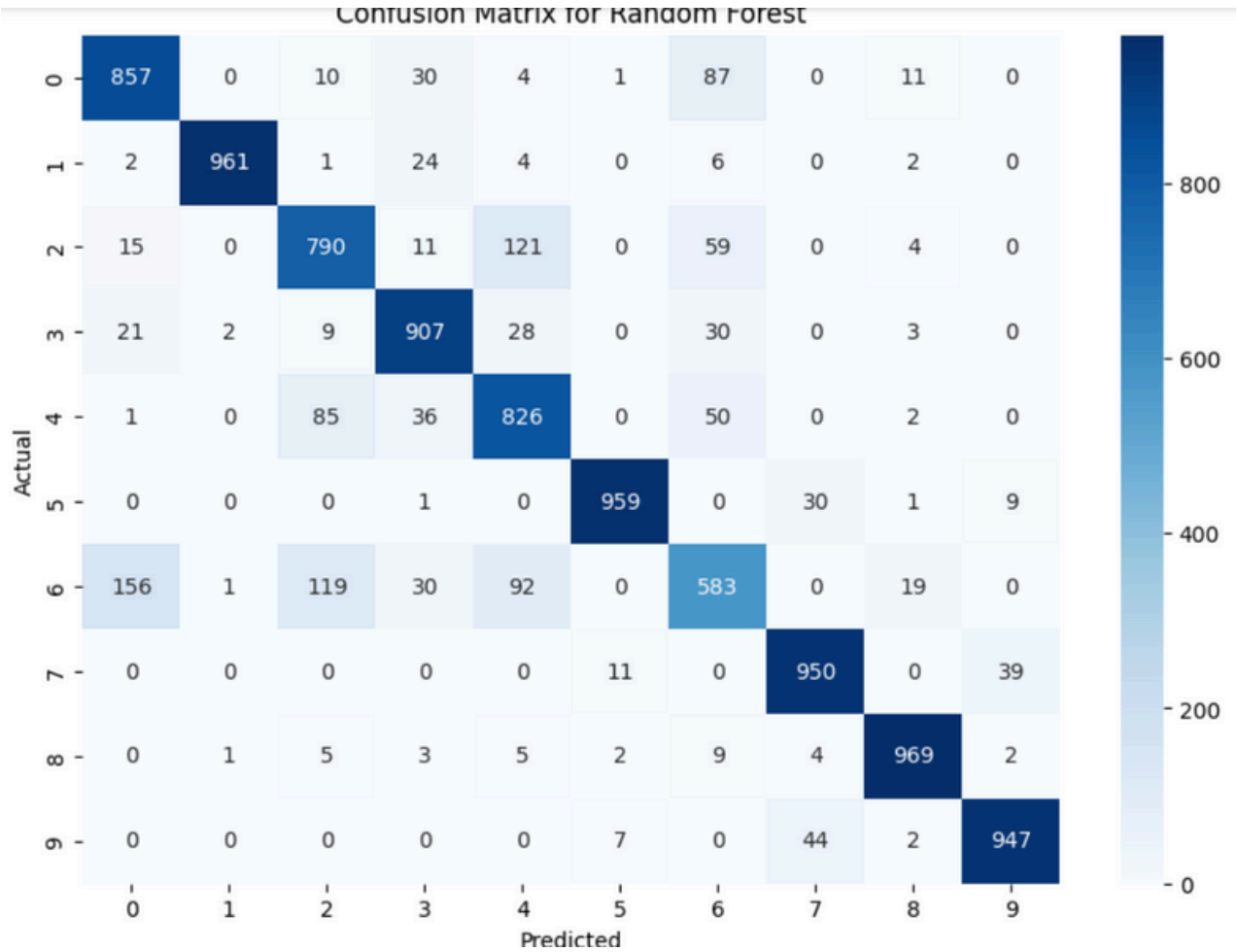
Random Forest Modeli Eğitimi ve Değerlendirmesi

```
Random Forest Modeli

random_forest = RandomForestClassifier()
accuracy, precision, recall, f1, roc_auc, cm = train_and_evaluate(random_forest, X_train_flat, y_train, X_test_flat, y_test, 'Random Forest')

if accuracy is not None:
    plt.figure(figsize=(10, 7))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
    plt.title('Confusion Matrix for Random Forest')
    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.show()
```

Random Forest | Accuracy: 0.8749, Precision: 0.8738, Recall: 0.8749, F1 Score: 0.8735, ROC AUC: 0.9895



Proje Adımları

3. ****Yapay Sinir Ağı (YSA) Modelinin Oluşturulması ve Eğitimi:**** Sequential model üzerinde yapılandırılan yapay sinir ağı, adam optimizer ve categorical_crossentropy loss fonksiyonu ile eğitilmiş. Eğitim sürecinde kayıp ve doğruluk değerleri takip edilmiş, sonuçlar epochlara göre grafikleştirilmiştir. Modelin performansı, accuracy, precision, recall, F1-score ve ROC AUC metrikleri kullanılarak değerlendirilmiş ve confusion matrix ile görselleştirilmiştir.

Yapay Sinir Ağı (YSA) Modeli Eğitimi ve Değerlendirilmesi

```
# Yapay Sinir Ağı (YSA) Modeli
model_nn = Sequential([
    Flatten(input_shape=(28, 28, 1)),
    Dense(128, activation='relu'),
    Dropout(0.2),
    Dense(10, activation='softmax')
])
model_nn.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Model Özeti
model_nn.summary()

# Modelin Eğitimi
history = model_nn.fit(X_train_resaped, y_train_cat, epochs=10, batch_size=128, validation_data=(X_test_resaped, y_test_cat), verbose=1)

# Eğitim Sonuçlarının Görselleştirilmesi
plt.figure(figsize=(14, 5))

# Kayıp Grafiği
plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], label='Training loss')
plt.plot(history.history['val_loss'], label='Validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.title('Training and Validation loss')
```

```
# Doğruluk Grafiği
plt.subplot(1, 2, 2)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.title('Training and Validation Accuracy')

plt.show()

# Model Değerlendirme
def train_and_evaluate_nn(model, X_train, y_train, X_test, y_test, model_name):
    y_pred_probs = model.predict(X_test)
    y_pred = np.argmax(y_pred_probs, axis=1)
    y_test_labels = np.argmax(y_test, axis=1)
    accuracy = accuracy_score(y_test_labels, y_pred)
    precision = precision_score(y_test_labels, y_pred, average='weighted')
    recall = recall_score(y_test_labels, y_pred, average='weighted')
    f1 = f1_score(y_test_labels, y_pred, average='weighted')
    roc_auc = roc_auc_score(y_test, y_pred_probs, multi_class='ovr')
    cm = confusion_matrix(y_test_labels, y_pred)

    print(f'{model_name} | Accuracy: {accuracy:.4f}, Precision: {precision:.4f}, Recall: {recall:.4f}, F1 Score: {f1:.4f}, ROC AUC: {roc_auc:.4f}')
    return accuracy, precision, recall, f1, roc_auc, cm

accuracy, precision, recall, f1, roc_auc, cm = train_and_evaluate_nn(model_nn, X_train_resaped, y_train_cat, X_test_resaped, y_test_cat, 'Neural Network')
```


Model Bilgileri

```
if accuracy is not None:
    plt.figure(figsize=(10, 7))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
    plt.title('Confusion Matrix for Neural Network')
    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.show()
```

Model: "sequential_1"

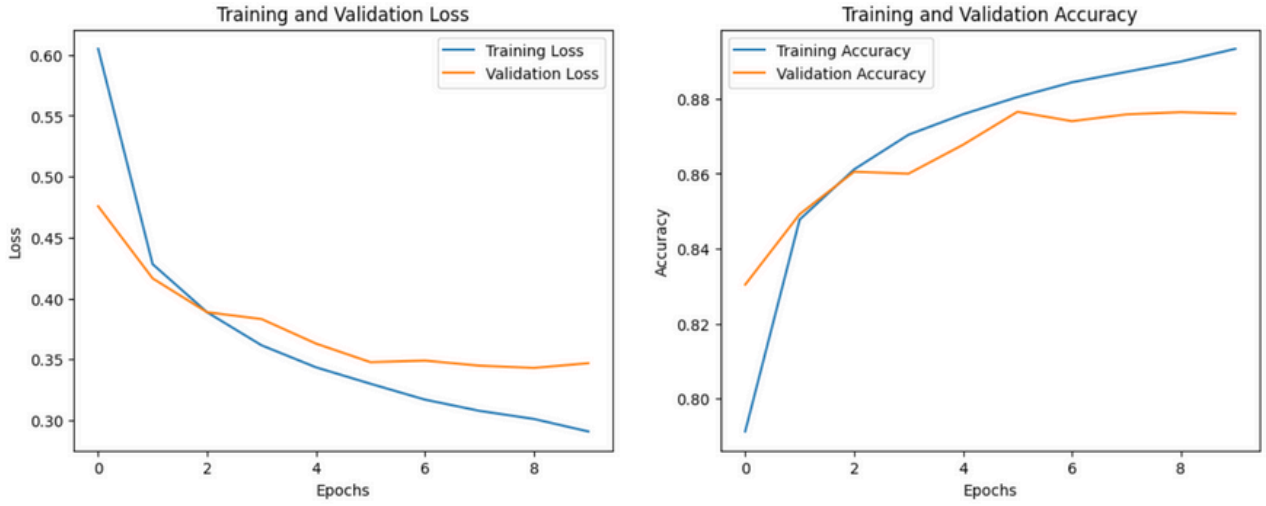
| Layer (type) | Output Shape | Param # |
|---------------------|--------------|---------|
| flatten_1 (Flatten) | (None, 784) | 0 |
| dense_2 (Dense) | (None, 128) | 100480 |
| dropout_1 (Dropout) | (None, 128) | 0 |
| dense_3 (Dense) | (None, 10) | 1290 |

=====
Total params: 101770 (397.54 KB)
Trainable params: 101770 (397.54 KB)
Non-trainable params: 0 (0.00 Byte)

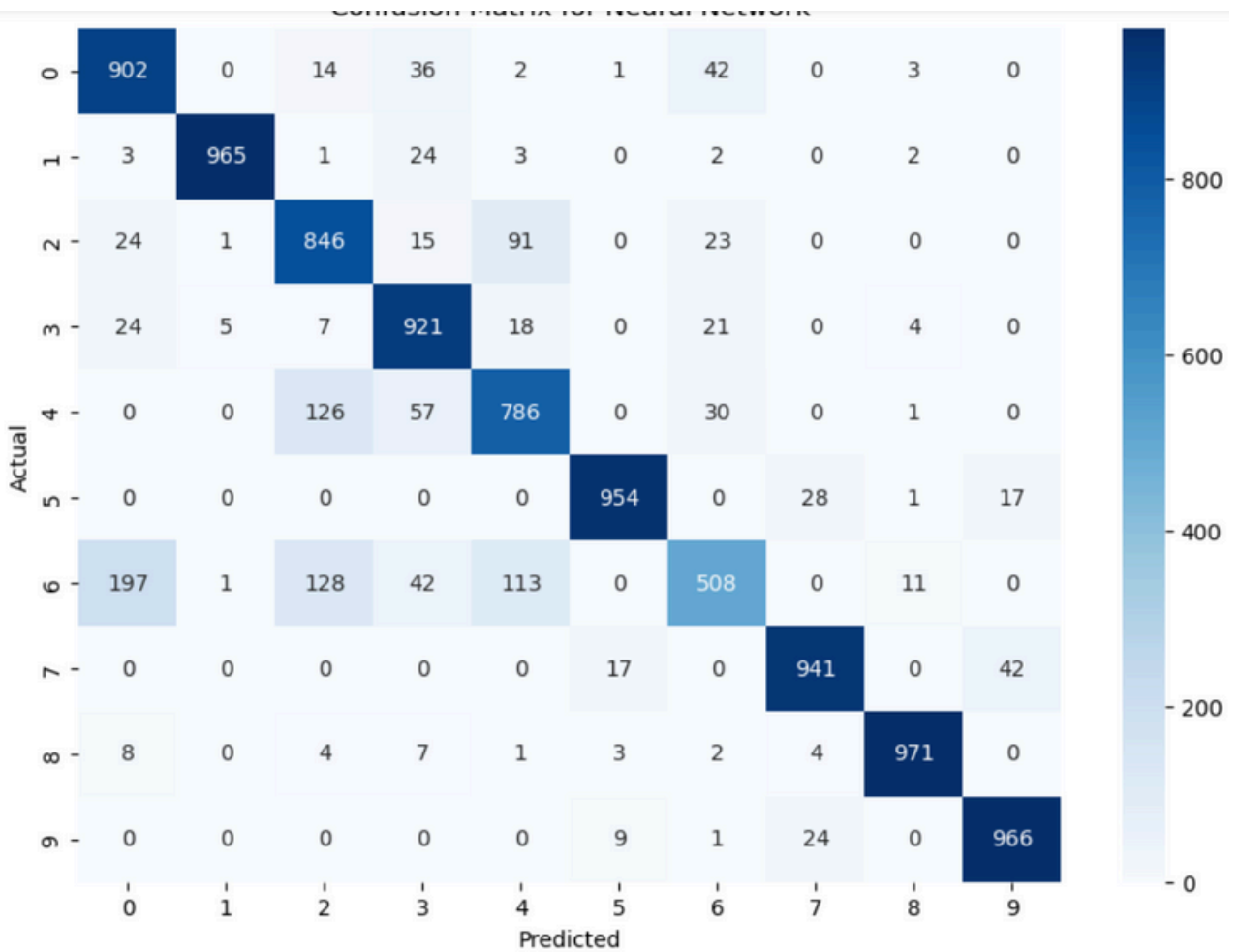
Metrik Sonuçları

```
Epoch 1/10
469/469 [=====] - 6s 9ms/step - loss: 0.6049 - accuracy: 0.7913 - val_loss: 0.4757 - val_accuracy: 0.8304
Epoch 2/10
469/469 [=====] - 2s 5ms/step - loss: 0.4282 - accuracy: 0.8478 - val_loss: 0.4165 - val_accuracy: 0.8492
Epoch 3/10
469/469 [=====] - 4s 8ms/step - loss: 0.3887 - accuracy: 0.8611 - val_loss: 0.3887 - val_accuracy: 0.8605
Epoch 4/10
469/469 [=====] - 3s 7ms/step - loss: 0.3616 - accuracy: 0.8704 - val_loss: 0.3832 - val_accuracy: 0.8600
Epoch 5/10
469/469 [=====] - 3s 6ms/step - loss: 0.3435 - accuracy: 0.8758 - val_loss: 0.3630 - val_accuracy: 0.8677
Epoch 6/10
469/469 [=====] - 2s 5ms/step - loss: 0.3301 - accuracy: 0.8804 - val_loss: 0.3478 - val_accuracy: 0.8765
Epoch 7/10
469/469 [=====] - 3s 6ms/step - loss: 0.3170 - accuracy: 0.8844 - val_loss: 0.3490 - val_accuracy: 0.8740
Epoch 8/10
469/469 [=====] - 3s 7ms/step - loss: 0.3078 - accuracy: 0.8871 - val_loss: 0.3449 - val_accuracy: 0.8758
Epoch 9/10
469/469 [=====] - 3s 7ms/step - loss: 0.3012 - accuracy: 0.8899 - val_loss: 0.3430 - val_accuracy: 0.8764
Epoch 10/10
469/469 [=====] - 2s 5ms/step - loss: 0.2909 - accuracy: 0.8933 - val_loss: 0.3469 - val_accuracy: 0.8760
```

Modelin eğitim sürecindeki kayıp ve doğruluk değerlerini içeren grafik sonuçları



Modelin karmaşıklık matrisi:



Kullanılan Veri Seti

Fashion MNIST veri seti, kıyafet ve aksesuar resimlerinden oluşan bir veri setidir. Bu veri seti, her biri 28x28 piksel boyutunda gri tonlamalı görüntülerden oluşur.

Toplamda 70,000 görüntü vardır; bunların 60,000'i eğitim ve 10,000'i test verisi olarak ayrılmıştır. Her görüntü, 0'dan 9'a kadar olan 10 farklı sınıftan birine aittir.

Analizler

KNN (K-Nearest Neighbors): Basit bir algoritma olmasına rağmen, sınıflandırma doğruluğu makul seviyedeydi ancak diğer metriklerde (precision, recall, f1-score) orta düzey performans gösterdi.

Decision Tree:: Aşırı uyum (overfitting) problemi nedeniyle test verisinde düşük performans gösterdi.

Random Forest: Birden fazla karar ağacını birleştirerek daha iyi genelleme yapar ve daha yüksek performans sağlar. Doğruluk ve diğer metriklerde iyi performans gösterdi, aşırı uyum problemini azalttı.

Kullanılan Yöntemler

Veri Ön İşleme

- Normalize Etme: Veriler, [0, 255] aralığında piksel değerlerine sahip olduğu için, bu değerler [0, 1] aralığına normalize edildi. Bu, modelin daha hızlı ve verimli öğrenmesini sağlar.
- Veri Düzleştirme: Makine öğrenmesi algoritmaları için veriler, 28x28 piksel boyutundaki görüntülerden 784 uzunluğunda düzleştirilmiş vektörlere dönüştürüldü.

Makine Öğrenmesi Modelleri

- KNN (K-Nearest Neighbors): Komşuluk tabanlı bir sınıflandırıcı olup, eğitim verilerini bellekte tutar ve tahmin yaparken en yakın komşuları kullanır.
- Decision Tree: Ağaç yapısı kullanarak veri setini sınıflandırır. Karar düğümleri ve yaprak düğümlerden oluşur.
- Random Forest: Birden fazla karar ağacı oluşturur ve bunların sonuçlarını birleştirerek daha doğru ve kararlı tahminler yapar.

Optimizasyon Fikirleri:

Veri Ön İşleme

- Daha İyi Normalizasyon Teknikleri: Z-score normalizasyonu gibi alternatif normalizasyon yöntemleri incelenebilir.

Model Optimizasyonu

- Hiperparametre Optimizasyonu: Grid search veya random search yöntemleri kullanılarak modellerin hiperparametreleri optimize edilebilir.
- Transfer Learning: Önceden eğitilmiş modeller kullanılarak transfer öğrenme yöntemleri uygulanabilir.
- Model Değerlendirme
- K-cross Validation: Modellerin performansını daha iyi değerlendirmek için k-fold çapraz doğrulama (cross-validation) kullanılabilir.
- Diğer Metrikler: Özellikle dengesiz veri setlerinde daha anlamlı sonuçlar veren metrikler (Matthews correlation coefficient, Cohen's kappa) incelenebilir.

Özet

Bu proje, katılımcılara Fashion MNIST veri seti üzerinde farklı makine öğrenmesi ve derin öğrenme modelleri ile çalışma imkanı sunar. Verilerin ön işlenmesi, modellerin oluşturulması ve eğitilmesi, sonuçların çeşitli metriklerle değerlendirilmesi gibi adımları içerir. Her modelin performansı karşılaştırılarak, hangi algoritmanın veri seti üzerinde daha iyi sonuç verdiği analiz edilir.

Katılımcılara çeşitli makine öğrenmesi ve derin öğrenme algoritmalarının nasıl kullanılacağını ve değerlendirileceğini öğretmektedir. Kullanılan Fashion MNIST veri seti, gerçek hayatta karşılaşılabilecek görüntü sınıflandırma problemlerine iyi bir örnektir. Makine öğrenmesi ve derin öğrenme modelleri arasındaki performans farkları ve bu farkların sebepleri üzerinde durmak, kullanıcılara model seçiminde dikkat edilmesi gereken noktalar hakkında değerli bilgiler sağlar.

Genel olarak, bu proje, yapay zeka ve makine öğrenmesi alanında temel bilgi ve deneyim kazanmak isteyenler için iyi bir başlangıç noktasıdır. İleride daha karmaşık ve büyük veri setleri üzerinde çalışarak, bu alandaki bilgi ve becerilerimizi daha da geliştirebiliriz.

KAYNAKÇA



youtube.com

<https://kadirguzel.medium.com/geri-yay%C4%B1l%C4%B1ml%C4%B1-%C3%A7ok-katmanl%C4%B1-yapay-sinir-a%C4%9Flar%C4%B1-1-47daa3856247>

<https://www.gtech.com.tr/yapay-sinir-aglari-ve-uygulamalari/https://medium.com/t%C3%BCrkiye/yeni-ba%C5%9Flayanlar-i%C3%A7in-makine-%C3%B6%C4%9Frenmesi-algoritmalar%C4%B1-ae22f794af2f>

<https://medium.com/machine-learning-t%C3%BCrkiye/makine-ogrenmesi-7cfbb3d859db>